RD&E CENTER

Technical Report

No. 13510

DESIGN OF A MICRO-CONTROLLER FOR ABSORBED POWER ANALYSIS

October 1990

By Alexander A. Reid
Gregory R. Hudas
U.S. Army Tank-Automotive Command
ATTN: AMSTA-RYA
Warren, MI 48397-5000

U.S. ARMY TANK-AUTOMOTIVE COMMAND
RESEARCH, DEVELOPMENT & ENGINEERING CENTER
Warren, Michigan 48397-5000

NOTICES

This report is not to be construed as an official Department of the Army position.

Mention of any trade names or manufacturers in this report shall not be construed as an official endorsement or approval of such products or companies by the U.S. Government.

Destroy this report when it is no longer needed. Do not return it to the originator.

REPORT DOCUMENTATION PAGE	1. REPORT NO.	2.	3. Recipient's Accession No.

4. Title and Subtitle
Design of a Micro-Controller for Absorbed Power Analysis

5. Report Date
October 1990

6.

7. Author(s)
Gregory R. Hudas and Alexander A. Reid

8. Performing Organization Rept. No.
13510

9. Performing Organization Name and Address
U.S. Army Tank-Automotive Command
System Simualtion and Technology Division (AMSTA-RY)
Warren, MI 48397-5000

10. Project/Task/Work Unit No.

11. Contract(C) or Grant(G) No.
(C)
(G)

12. Sponsoring Organization Name and Address
U.S. Army Tank-Automotive Command
System Simulation and Technology Division (AMSTA-RY)

13. Type of Report & Period Covered
1/90-4/90

14.

15. Supplementary Notes

16. Abstract (Limit: 200 words)

The U.S. Army uses "absorbed power" to measure the level of discomfort of an occupant riding in a vehicle. Average absorbed power can be calculated if the time history of acceleration at a given point (for example) the driver's seat is known. The smaller the absorbed power is for a given vehicle, running over a specific profile at a certain speed, the better vehicle ride is. There is a need for Application Specific Integrated Circuit (ASIC) chips and microcontrollers in order to construct an absorbed power measuring instrument.

This report describes the basic digital logic design and simulation results of an ASIC developed to calculate "absorbed power".

17. Document Analysis a. Descriptors

Simulation, Absorbed Power, Micro-controller, Transfer Function, Bus/Register
External memory, Instruction, Floating point, clock cycle

b. Identifiers/Open-Ended Terms

PC-MATLAB, VAX, AHPL Simulator (HPSIM2)

c. COSATI Field/Group

18. Availability Statement	19. Security Class (This Report)	21. No. of Pages
Approved for Public Release: Distribution is Unlimited.	Unclassified	40
	20. Security Class (This Page) Unclassified	22. Price

TABLE OF CONTENTS

Section		Page
1.0	INTRODUCTION	7
2.0	OBJECTIVE	7
3.0	CONCLUSION	7
4.0	RECOMMENDATIONS	8
5.0	DISCUSSION	8
5.1	Absorbed Power	8
5.2	Bus/Register Description	9
5.3	Instruction Set	14
5.4	Use of the Micro-Controller	18
5.5	Floating Point	18
5.6	External Memory	19
5.7	Simulation	19
LIST OF REFERENCES		23
ADDENDUM	HPSIM Description of ASIC	25
DISTRIBUTION LIST		DIST-1

LIST OF ILLUSTRATIONS

Figure	Title	Page
5-1.	Side-Side Absorbed Power Bode Plot	9
5-2.	Fore-Aft Absorbed Power Bode Plot	10
5-3.	Vertical Absorbed Power Bode Plot	10
5-4.	Bus Structure of Micro-Controller	12
5-5.	Inputs/Outputs for Micro-Controller	13
5-6.	CSR Register	14
5-7.	Karnaugh Map of Instruction Set	14
5-8.	CMP Instruction	15
5-9.	OUT Instruction	15
5-10.	GO Instruction	16
5-11.	LD Instruction	16
5-12.	ADD Instruction	17
5-13.	BRx Instruction	17
5-14.	Floating Point Format	18
5-15.	Module Layout	20

1.0. INTRODUCTION

Currently, the U.S. Army is engaged in the studies of human vibration and its impact on the design of combat and tactical vehicles. Because of this, standards have been developed to assist researchers in observing the effects of the human body undergoing vibrating forces under certain conditions. These standards are important in the development of vehicle subsystems such as seats, safety restraints, and suspension systems. To make effective use of these standards comes the need to design logic for Application Specific Integrated Circuit (ASIC) chips and micro-controllers in order to calculate and compute data in a very quick manner.

The analysis used in this report was prepared by the authors at the U.S. Army Tank-Automotive Command, Analytical and Physical Simulation Branch, in conjunction with the requirements for a final class project in the class "Design of Digital Systems (ECE 666)" taken during the winter 1990 semester at Wayne State University, Detroit, Michigan.

2.0. OBJECTIVES

The objectives of the following paper are to give the basic digital logic design and the simulation results of an ASIC developed to calculate a standard called "absorbed power". The main topics to be discussed in this report are as follows:

- Introduction to the theory of absorbed power.
- Processor Architecture.
- The HPSIM simulation model and results.

It must be stressed that the purpose of this report is to present an ASIC logic design developed by the authors using HPSIM software created by the University of Arizona.

3.0. CONCLUSION

The complicated and tedious task of calculating absorbed power can be simplified greatly with the construction and use of this integrated circuit chip. With the addition of analog-to-digital (A/D) converters onto the chip, the only additional hardware needed to calculate absorbed power is accelerometers. This is in respect to the current practice of using accelerometers, a recording medium to gather data and a computer for the computation of the absorbed power.

4.0. RECOMMENDATIONS

While the layout of the logic for this chip is complete, more work should be done to verify this before the costly process of integrated chip manufacturing is carried out. It may be a good idea to also include the International Standards Organization (ISO) ride-level standards on this chip.

5.0. DISCUSSION

5.1. Absorbed Power

Absorbed power is described as the power a human body will absorb when exposed to a vibrational environment. Absorbed power is a military standard which is used to test vehicle suspension systems. A person will only tolerate up to six watts of absorbed power before he/she will slow down the vehicle to reduce this level.

To calculate absorbed power, the three linear accelerations must be measured (x,y and z) in ft/sec^2. These are used as inputs to the absorbed power transfer functions and the outputs are then squared, averaged and summed together to create total absorbed power. The transfer functions are listed here in the continuous time domain:

VERTICAL: $$\frac{15.453s(s+5.0)(s^2+28.3s+2800.0)(s^2+105.0s+7570.0)}{(s+6.0)(s^2+29.8s+1000.0)(s^2+39.1s+3800.0)(s^2+125.0s+5180.0)}$$

FORE-AFT: $$\frac{209.0s(s+110.0)}{(s^2+17.6s+125.0)(s^2+110.0s+1300.0)}$$

SIDE-SIDE: $$\frac{478.0s(s+130.0)(s^2+11.2s+60.0)(s^2+14.2s+260.0)}{(s^2+3.33s+17.0)(s^2+5.5s+140.0)(s^2-44.0s+900.0)(s^2+255.0s-2500.0)}$$

To make optimum use of a digital controller, these transfer functions have been converted over to the discrete domain using PC-MATLAB. The digital transfer functions have a sampling period of 0.005 seconds and are as follows:

VERTICAL: $$\frac{0.029881z^{-7}-0.19901z^{-6}+0.59394z^{-5}-0.99171z^{-4}+0.96532z^{-3}-0.51607z^{-2}+0.11765z^{-1}}{-0.13547z^{-7}+1.1952z^{-6}-4.6053z^{-5}-10.023z^{-4}-13.315z^{-3}+10.811z^{-2}-4.973z^{-1}+1.0}$$

FORE-AFT: $$\frac{3.0516z^{-4}-8.9755E-3z^{-3}-3.8404E-3z^{-2}+9.7642E-3z^{-1}}{0.27915z^{-4}-1.6606z^{-3}+3.4643z^{-2}-3.082z^{-1}+1.0}$$

SIDE-SIDE:

$$\frac{0.0022017z^{-8}-0.016133z^{-7}+0.035796z^{-6}-0.010529z^{-5}-0.07096z^{-4}+0.11392z^{-3}-0.07043z^{-2}+0.016138z^{-1}}{0.046037z^{-8}-0.88328z^{-7}+5.2066z^{-6}-15.376z^{-5}+26.491z^{-4}-27.999z^{-3}+17.975z^{-2}-6.4596z^{-1}+1.0}$$

The user can load the scale-factor registers (vertical scale factor: VSF; fore-aft scale factor: FSF; and the side-side scale factor: SSF) with a value (ft/sec^2/volt) to convert the input value to the correct acceleration. Bode plots for these transfer functions are presented in Figures 5-1, 5-2, and 5-3.

5.2 Bus/Register Description

The basic block diagram (Figures 5-4 and 5-5) consists of several registers and buses. The operating registers in which the user has access through instructions are the accumulator (AC), index register (IX), and program counter (PC). Two non-operating registers are the instruction register (IR) and the memory data register (MD). A block of computational registers called RAMREG consists of both operating registers and non-operating registers which will be discussed in the next paragraph. To have access to memory, the buses ADBUS and DBUS are implemented. Both ABUS and BBUS provide interconnections between the registers and the inputs to the Arithmetic Logical Unit (ALU), while OBUS serves as a connection between the ALU outputs and register inputs.

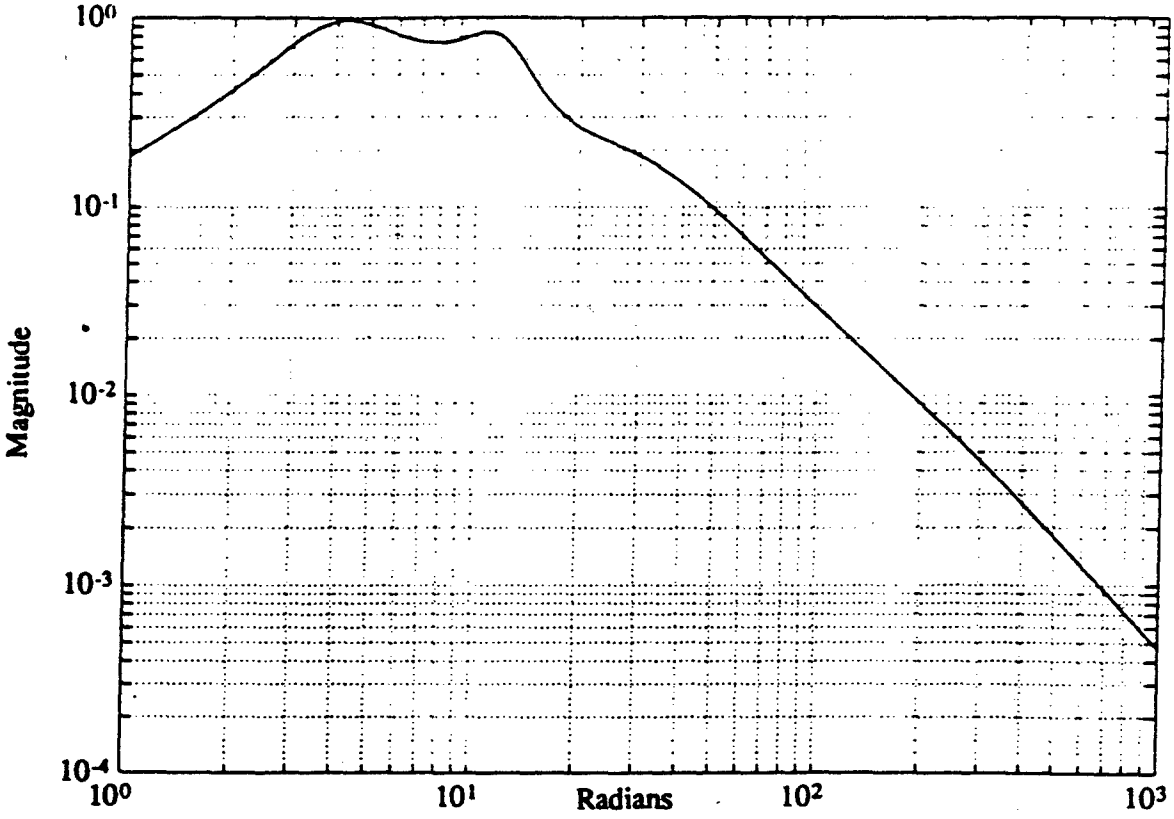


FIGURE 5-1. Side-Side Absorbed Power Bode Plot

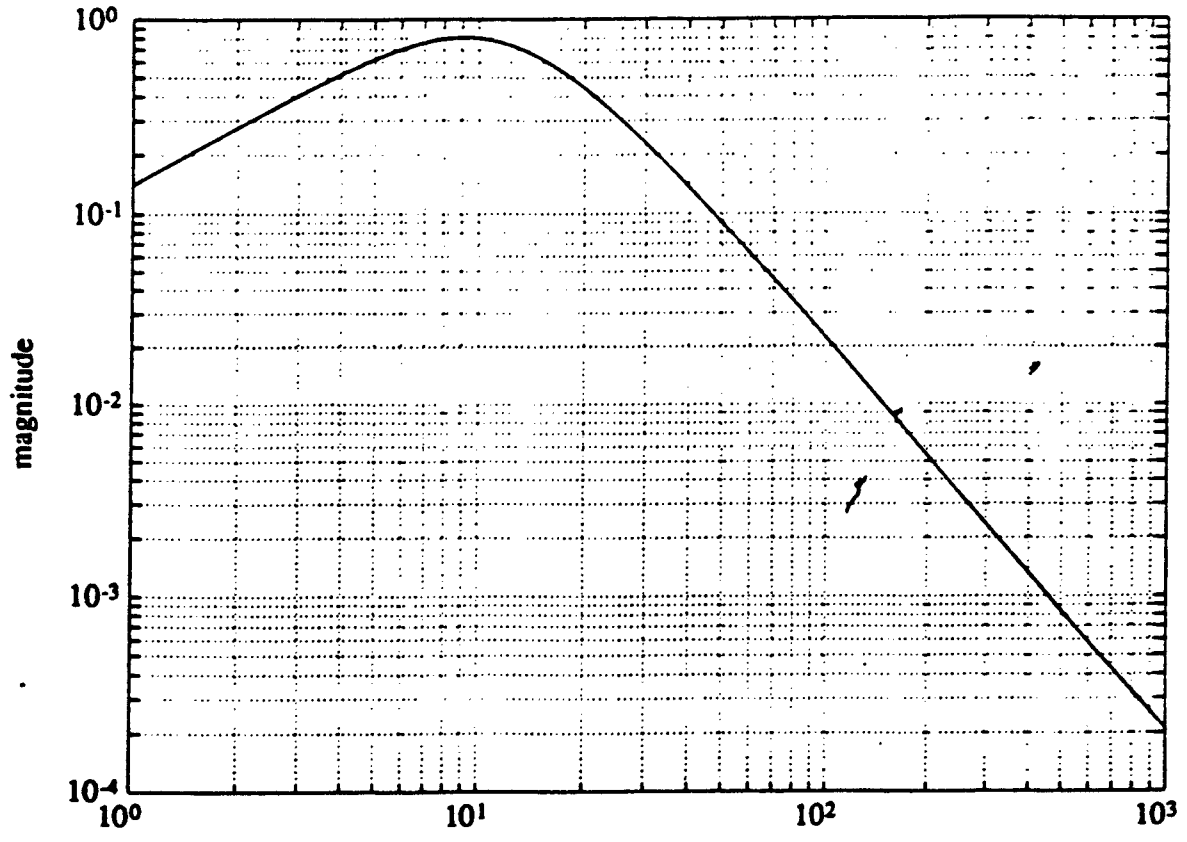


FIGURE 5-2. Fore-Aft Absorbed Power Bode Plot

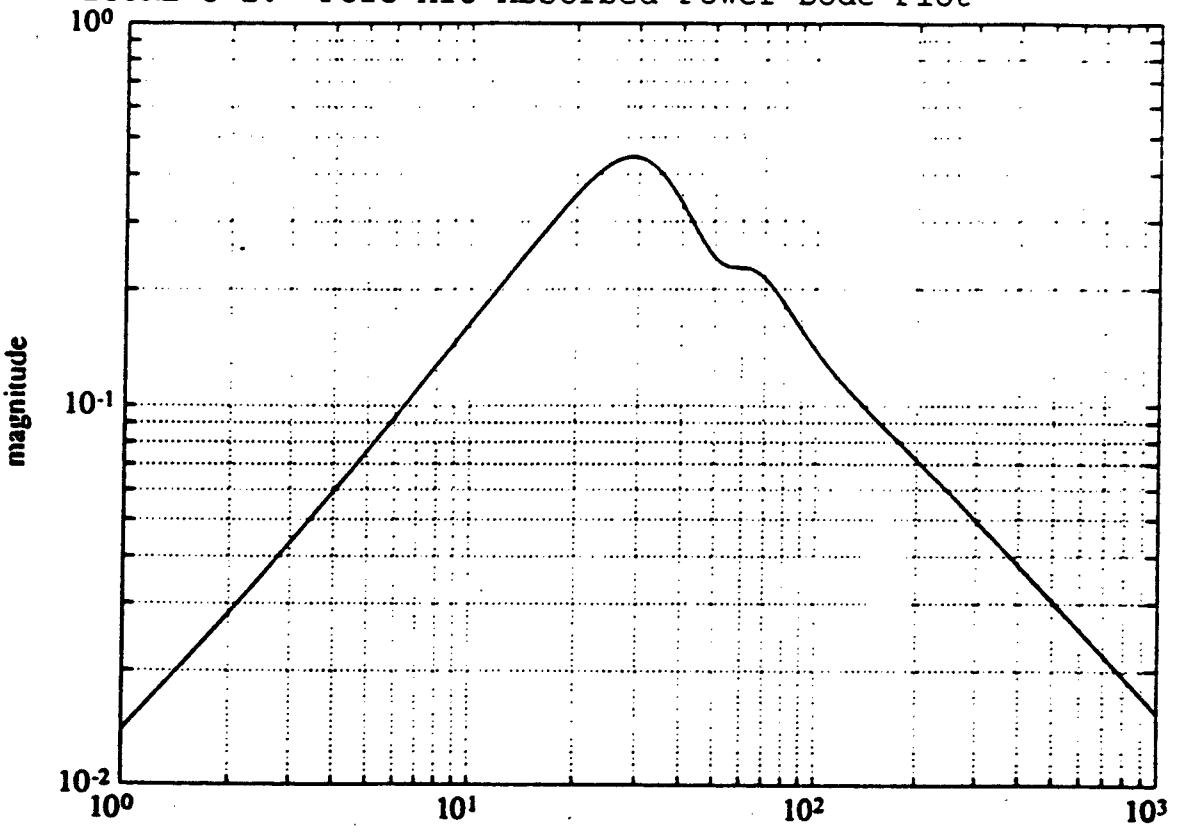


FIGURE 5-3. Vertical Absorbed Power Bode Plot

The RAMREG block, which plays an important role in the computations regarding the absorbed power transfer functions, is made up of the following registers:

TIME[28]: A 28 bit nonoperating register which keeps a running total of time. It will be used in averaging the result.

COUNT[28]: A nonoperating register used as a 28 bit decrementing counter.

TEMP[28]: Nonoperating register which is used for temporary storage during floating point operations.

VERTICAL<16>[28]: A block of nonoperating registers used in the time dependent computations of vertical absorbed power. These registers contain the time delay values, running sum values, and total average values during the course of the calculations.

FORE-AFT<10>[28]: A block of nonoperating registers used in the time dependent computations of fore-aft absorbed power.

SIDE-SIDE<18>[28]: A block on nonoperating registers used in the time dependent computations of side-side absorbed power.

SMP[28]: A 28 bit operating register in which the user inputs the number of acceleration data samples.

CSR[8]: An 8 bit control/status register (Figure 5-6). The user can observe the floating point error (FLERR), negative flag (NFF), overflow flag (VFF), carry flag (CFF), and zero flag (ZFF) by reading bits 0 through 4. Bits 5 through 7 serve as on/off switches (1 = on) for the three absorbed power calculations.

Referring to Figure 5-5, the system is comprised of four EXINPUTS and four possible OUTPUTS. The inputs VIN, FIN, and SIN represent the vertical fore-aft and side-side acceleration inputs respectively. The two possible output formats are digital and analog. To obtain analog outputs, FOREOUT, SIDEOUT, and VERTOUT are sent though 3 separate d/a converters. To obtain the digital output, the degree of freedom selected is output onto a 28 bit bus called OUTPUTBUS.

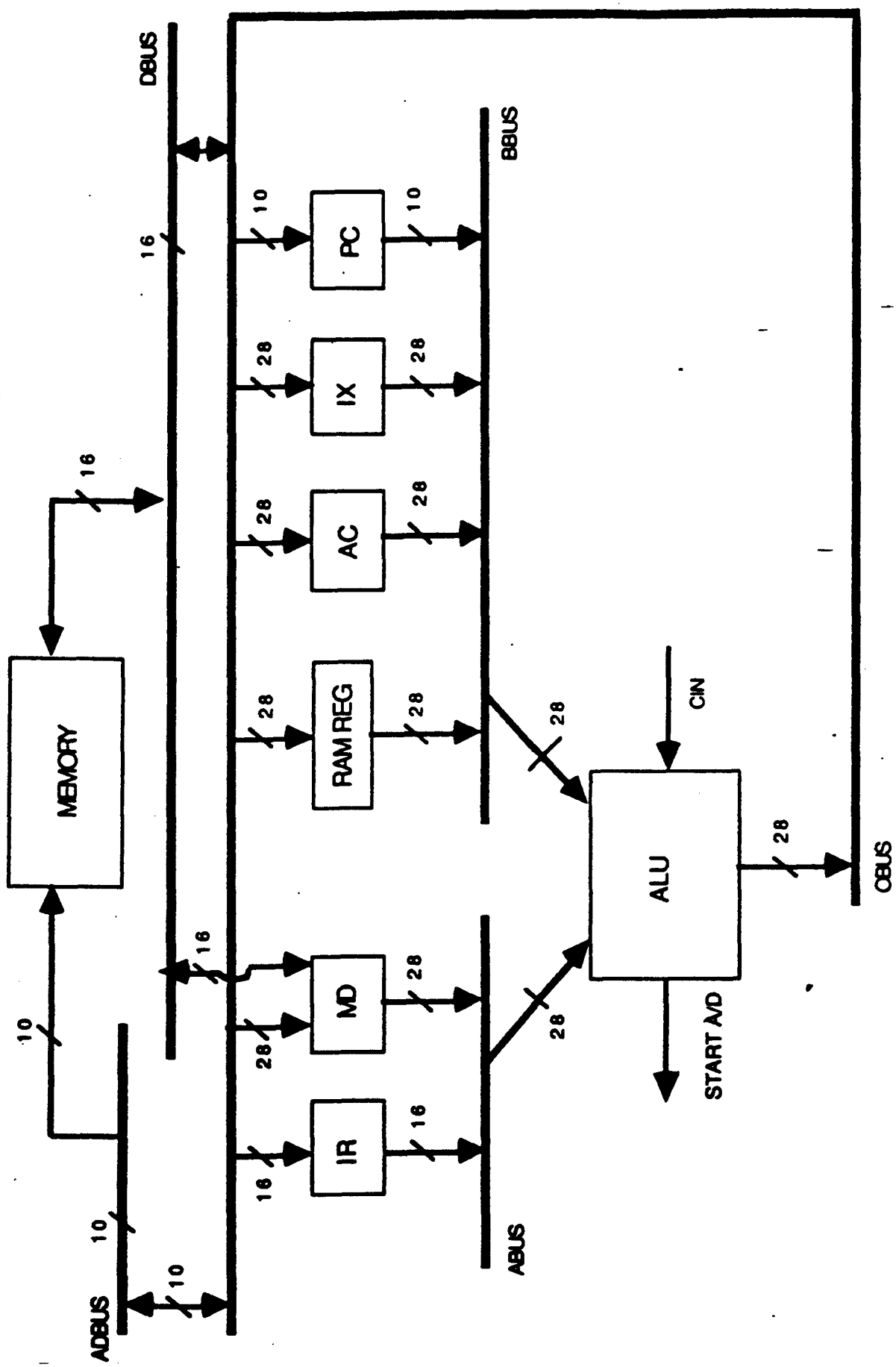


FIGURE 5-4. Bus Structure of Micro-Controller

OUTPUTS:

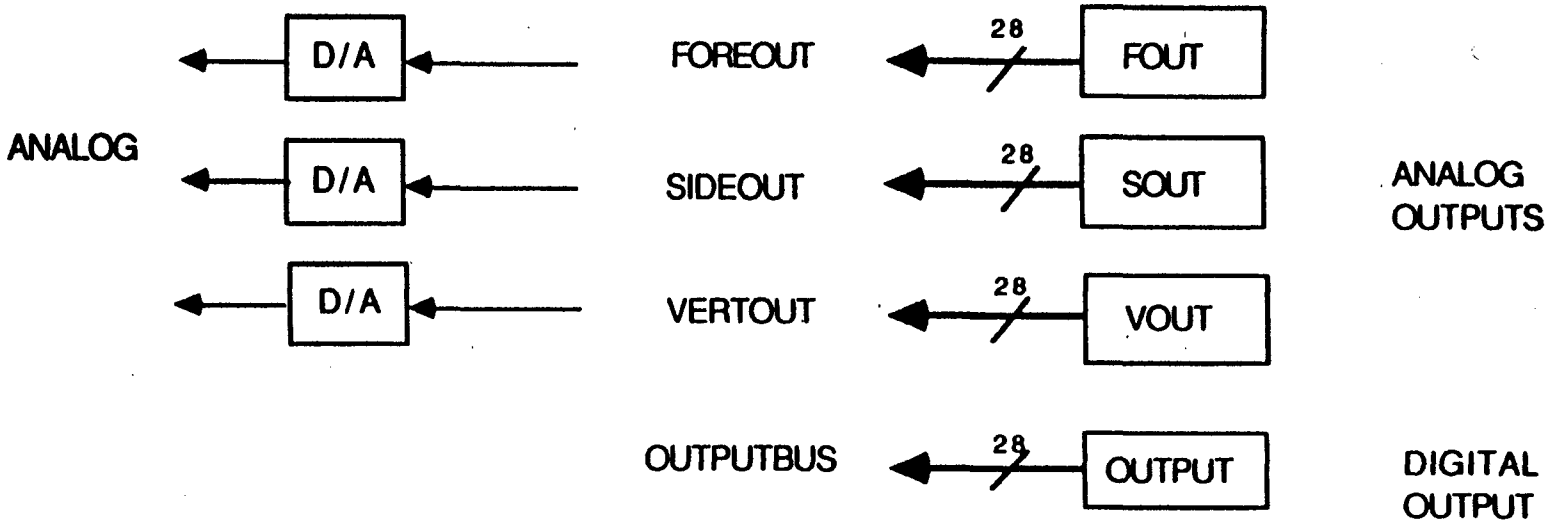
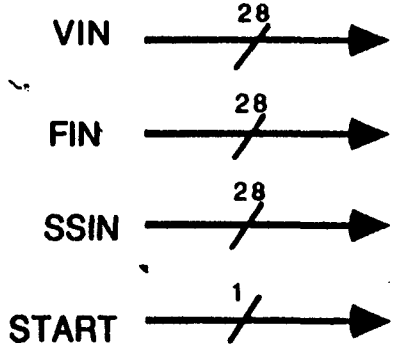
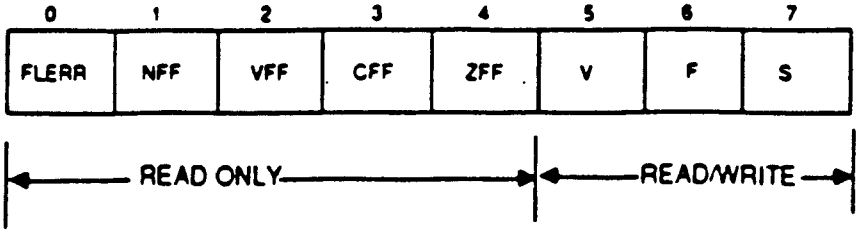


FIGURE 5-5. Inputs/Outputs for Micro-Controller

EXINPUTS:





NOTE: V - Vertical (1 is on)
F - Fore-aft
S - Side-Side

FIGURE 5-6. (CSR Register)

5.3. Instruction Set

In this system, the instruction cycle consists of both the fetch operation and the execute operation. Because time is very crucial in the computation of absorbed power using this ASIC design, the instruction cycle was optimized by grouping related instructions together using a Karnaugh map (Figure 5-7). For instance, notice that the Branch commands contain various common

IR[0:1]

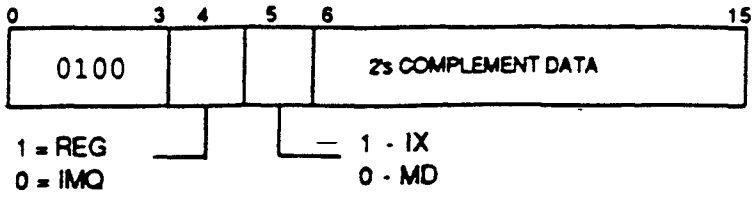
IR[2:3]	00	01	11	10
00	ADD	CMP	X	BRZ
01	X	GO	X	BRV
11	X	X	X	BRN
10	OUT	LD	X	BRA

FIGURE 5-7. (Karnaugh Map of Instruction Set)

features and have been grouped with the same IR[0:1]. In fact, by using the "don't cares" within the K-map will reveal a reduced minterm IR[0].

The instruction set is comprised of nine instructions to serve as an interface between the user and the ASIC. They are discussed in full in the following paragraphs. In all cases, IR[0:3] is the opcode of the instruction.

The **CMP** instruction (Figure 5-8) compares either a register (IX or MD) or immediate 2's complement data to the contents of the

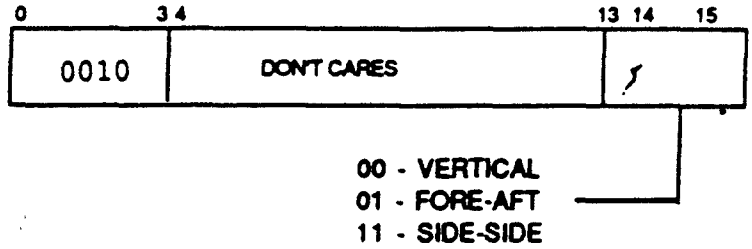


NOTE: If bit 4 = 1 (REG), bit 5 specifies the register to CMP to AC.
If bit 4 = 0 (IMQ), bits 5 - 15 are 2's comp data.

FIGURE 5-8. (CMP Instruction)

accumulator (AC) and sets the status bits of the CSR accordingly without modifying the contents of AC.

The **OUT** instruction (Figure 5-9) simply outputs an assigned



NOTE: This instruction is used to assign a given digital output port if needed.

FIGURE 5-9. (OUT Instruction)

data type to the output bus (OUTPUTBUS). As shown in the figure, bits 14 and 15 of the IR will determine which data is to be sent to OUTPUTBUS.

The **GO** instruction (Figure 5-10) is used to begin the entire

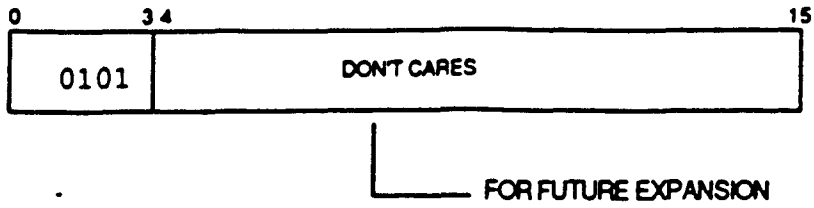


FIGURE 5-10. GO Instruction

process. When this instruction is implemented, first, the CSR is checked to see which input accelerations are to be used for the computation of absorbed power. Next, all required registers are initialized. Finally, the time dependent calculations of absorbed power are made based on the specified input accelerations.

The **LD** instruction loads the contents of a specified register or immediate data into a specified destination register (Figure 5-11). IR[4:5] gives the destination while IR[14:15] gives the source. IR[13] specifies whether the

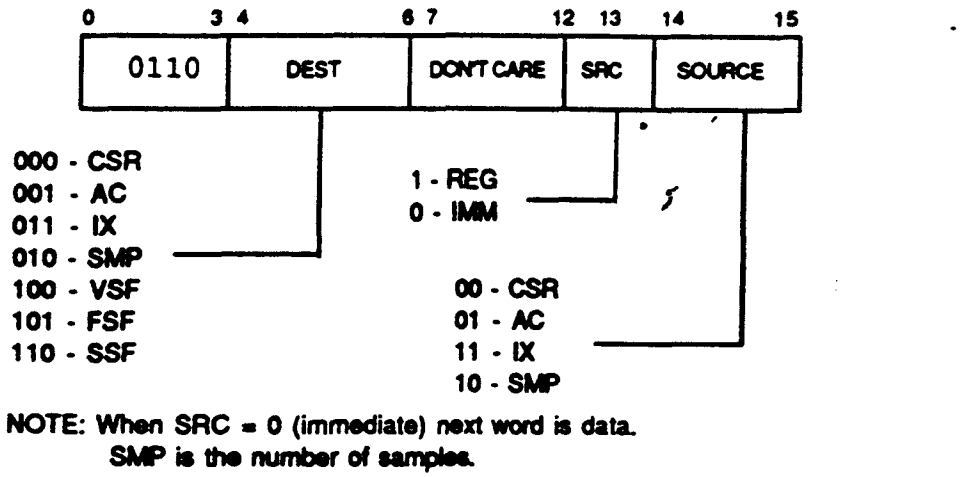


FIGURE 5-11. LD Instruction

source is immediate or register data. If immediate, the next word (in 2's complement form) is fetched and read. It is with this command that the scale factors are loaded into VSF, FSF, and SSF.

The **ADD** instruction (Figure 5-12) simply adds a 2's complement number to either AC or IX as specified by IR[4].

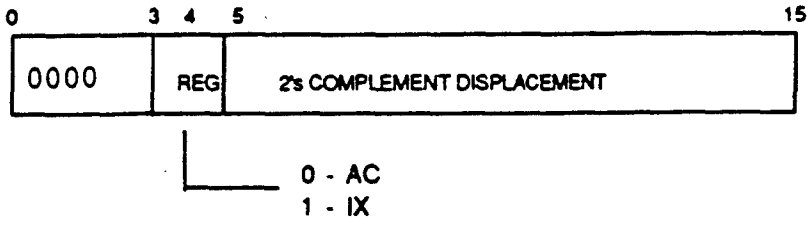


FIGURE 5-12. ADD Instruction

If AC is selected, the result is a standard ADD operation. The register IX will be specified if indexed addressing is needed.

Finally, the **BRx** instructions (Figure 5-13) are made up of

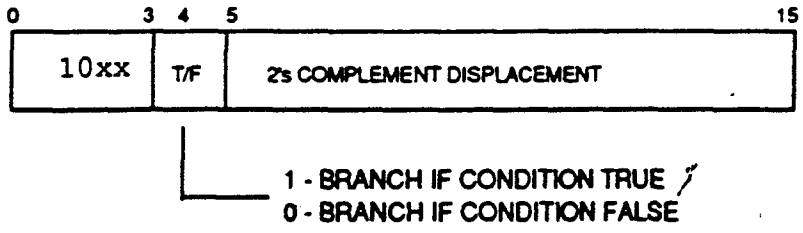


FIGURE 5-13. BRx Instructions

three conditional branch instructions (BRZ, BRN, BRV) and one unconditional branch instruction (BRA) depending on the opcode selected. The 11 bit displacement (IR[5:15]) allow for 2's complement number permitting a branch range of 1K backward or forward.

All nine of these instruction operations will be seen later in the AHPL model.

5.4. Use of the Micro-Controller

Before the 'GO' command can be issued, the user must first set up the micro-controller for use. The 'SMP' register must be loaded with an unsigned integer representing the number of samples to take. Sampling is done at 200 Hz, so the micro-controller will calculate absorbed power for SMP/200 seconds. The scaling factors must be loaded into the registers for vertical scaling (VSF), side-side scaling (SSF) and fore-aft scaling (FSF). These values are in the floating point form described in the next section. Once this is done, the micro-controller is now ready to perform absorbed power analysis.

5.5. Floating Point

To obtain the accuracy and wide range of numerals used in the absorbed power computations, floating point notation was used (see Figure 5-14). The only difference between this format and the IEEE floating point format is the length of the mantissa is 19 bits instead of 23 bits. All of the computational registers used for calculation of the absorbed power are floating point registers. To perform mathematical operations on these registers, three external modules were created, one each for multiplication, division and addition (see Figure 5-15 for the layout of the modules used in the this project).

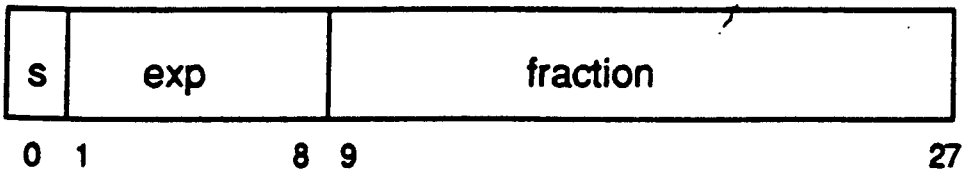


FIGURE 5-14. (Floating Point Format)

Floating-point addition is performed by the module 'FLADD'. The module expects the input numbers on the ABUS and BBUS. It waits for the line 'FLADDGO' to go high, reads the values off of ABUS and BBUS and performs the floating point operations. The exponents are first normalized, then the mantissas are added. The exponent is then adjusted (if the sum was greater than ten), the result is placed on OBUS, and the ACK bus is set high to let the main program know the addition is complete.

Floating point multiplication is performed by the module 'FLMUL'. This module also expects the inputs on ABUS and BBUS and begins operation when the 'FLMULGO' line is set high. The exponents are added and the mantissas are multiplied. The exponent is then adjusted (if the multiplicand is greater than ten), the result is placed on OBUS, and the ACK bus is set high to let the main program know the multiplication is complete.

Floating point division is similar to the multiplication and is performed by the external module 'FLDIV'. When the line 'FLDIV' is set high by the main module, the dividend is read off of the ABUS and the divisor is read off the BBUS. If the dividend is zero, the result is zero and it branches to the end of the module. If the divisor is zero, the result is infinity, the 'FLERROR' flag is set, and again the module ends. With valid numbers, the exponents are subtracted, the mantissas divided and the exponent adjusted. The result is placed on the OBUS and the ACK bus is set high to let the main program know the division is complete.

5.6. External Memory

Reading and writing to external memory is taken care of in the external module 'EXTMEM'. There are 'READ', 'WRITE', 'ADBUS' and 'DBUS' buses used as inputs to the memory module and 'ACK' as an output. To read from memory, set READ high, load ADBUS with the address location. The module will load DBUS with the data work and set ACK high signalling that the read operation is over. To write to the memory, set WRITE high, load ADBUS with the address and DBUS with the value to be written to memory. The module will write the value to memory and set ACK high to signify completion.

The memory module contains two NULL states. While not necessary, these were included to simulate slow (and inexpensive) memory which can and should be used with this chip. There is no reason to buy expensive fast memory when the sampling rate is only 200 Hz.

5.7. Simulation

The simulation results are presented in full in Addendum. The instruction fetch was done in lines 2-3. Decoding is performed in lines 4-7. Execution is broken down into the following segments:

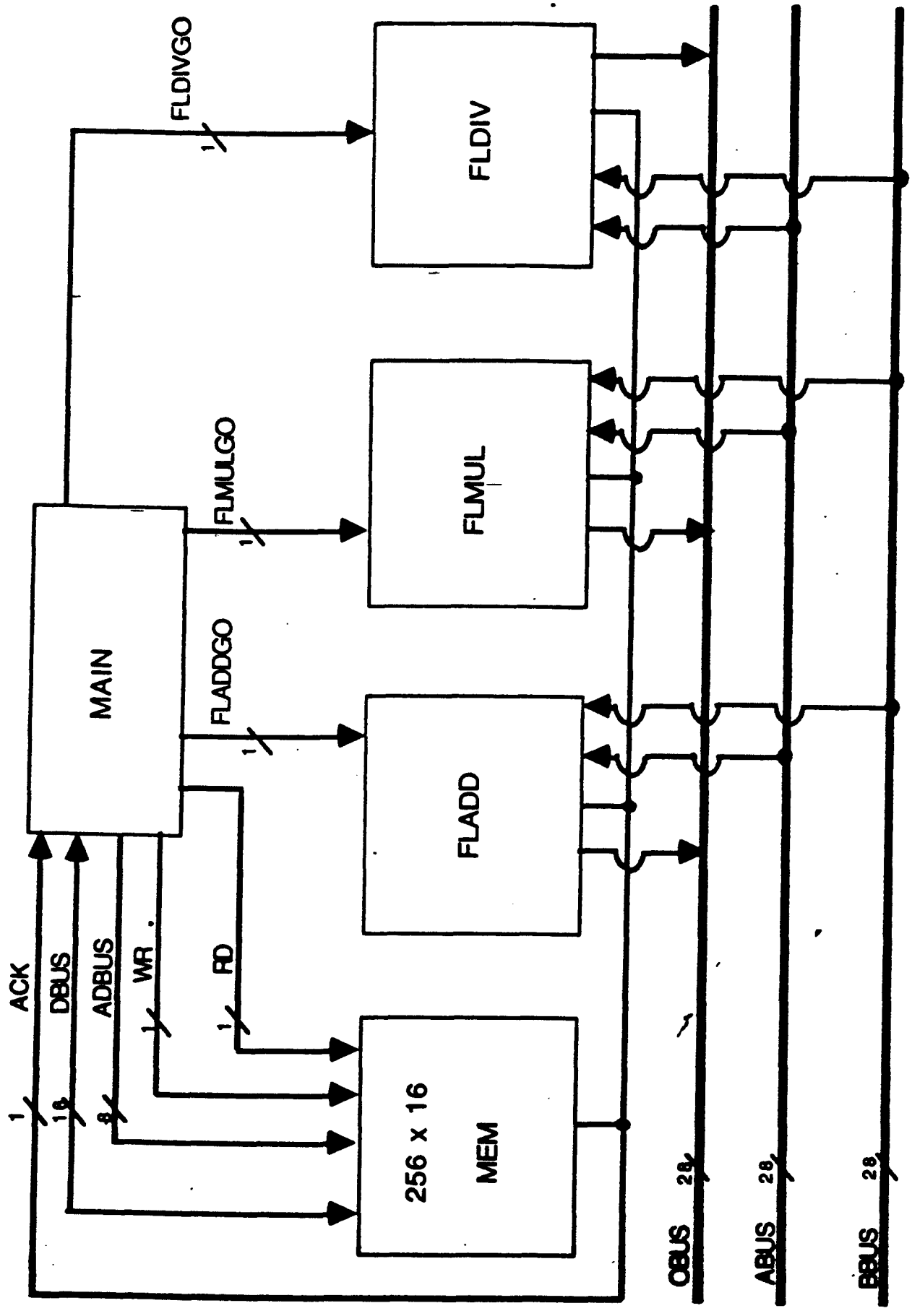


FIGURE 5-15. Module Layout

COMMAND	LINES OF CODE
ADD	8-9
BRANCH's	10-14
GO	15-138
LD	139-141
OUT	142-151
CMP-	152-156

The 'GO' command consists of three parts: computing vertical absorbed power (steps 18-61); side-side absorbed power (88-138); and fore-aft absorbed power (steps 62-87). The execution of each particular computation depends on the setting of the correct bits in the CSR (control-status register).The simulation was performed on a micro-VAX II using HPSIM2. There were no errors and all commands performed correctly. Referring to the simulation results (beginning on page A-15), a START command was issued on clock cycle 4. The first instruction was ADD AC,#F. This was performed during clock cycles 5-12. Note the value of 'F in AC at clock cycle 13.

The next instruction was LD IX,#5. This was performed during clock cycles 13-20. Note the value of IX at clock cycle 21. The following instruction is LD AC,'FFFF during clock cycles 21-35. The value of AC is 'FFFF at clock cycle 36. Finally, a load from register to register is presented as LD IX,AC during clock cycles 36-47. At clock cycle 48 both AC and IX have the value of 'FFFF.

A branch always was performed during clock cycles 48-55. The branch command was BRA #5. Note that PC is incremented from '06 to '0B at clock cycle 57. Another add was performed as ADD AC,'B during clock cycles 57-63. Register AC is properly increased to '10009. During clock cycles 64-78, the instruction LD AC,#'0A is performed. Clock cycle 79 shows the new value of AC to be '0A.

A compare is done during clock cycles 79-87. The CMP AC,#'0A does set the zero flag (clock cycle 88) as would be expected. The rest of the simulation begins with the GO command. This is meaningless as the data into the acceleration input registers have no value. Nonetheless, the command is properly executing.

To permit accuracy, all computational registers are 28 bits long (they would normally be 32 bits except for the compilation limitations of HPSIM). The user is provided with 256 words of 16 bit length.

LIST OF REFERENCES

1) "Digital Systems Hardware Organization and Design", Frederick J. Hill, Gerald R. Peterson, John Wiley & Sons, Inc., 1987.

2) "User Manual for AHPL Simulator (HPSIM2) AHPL Compiler (HPCOM)", Z. Navabi, R. Swanson, F.J. Hill, University of Arizona.

ADDENDUM

HPSIM DESCRIPTION OF ASIC

```
AHPLMODULE: ABPOWER

     EXINPUTS: SELECT[2].

     EXBUSES: DATAOUT[28].

     BUSES: ABUS[28]; BBUS[28]; OBUS[28]; CIN.

     MEMORY: RAM<65536>[28];MA[16];IR[28];MD[28];AC[28];IX[16];PC[16];CSR[28];
             SMP[28];TEMP[28];COUNT[28];VIN[28];SIN[28];FIN[28];TIME[28];
             VI7[28];VO7[28];VI6[28];VO6[28];VI5[28];VO5[28];VI4[28];VO4[28];
             VI3[28];VO3[28];VI2[28];VO2[28];VOUT[28];SI8[28];SO8[28];
             SI7[28];SO7[28];SI6[28];SO6[28];SI5[28];SO5[28];SI4[28];SO4[28];
             SI3[28];SO3[28];SI2[28];SO2[28];SOUT[28];FI4[28];FO4[28];FI3[28];
             FO3[28];FI2[28];FO2[28];FOUT[28];VSUM[28];VAP[28];SSUM[28];
             SAP[28];FSUM[28];FAP[28];OUTPUT[28];SELECT[2].

     LABELS: ZFF = CSR[25]; CFF = CSR[26]; NFF = CSR[27]; VFF=CSR[28].

     CLUNITS: FLADD[28] (ABUS;BBUS) ; FLDIV[28] (ABUS;BBUS) ;
              FMULT[28] (ABUS;BBUS) ; INC[28] (ABUS);
              DEC[28] (BBUS); BUSFN[28] (M; DCD); DCD[16] (MA) ;
              ADD0[1](ABUS;BBUS;CIN); ADD1[2](ABUS;BBUS;CIN);
              ADD28[29](ABUS;BBUS;CIN).

1.   OBUS = 28$256; PC <= OBUS; MA <= OBUS.

2.   MD <= BUSFN(M;DCD(MA)); PC <= INC(PC).

3.   ABUS = MD; OBUS = ABUS; IR <= OBUS.

4.   => (IR[0]) / (10).

5.   NO DELAY
     => (IR[3]) / (15).

6.   NO DELAY
     => (IR[2]) / (139).

7.   NO DELAY
     => (IR[1]) / (151).

8.   ABUS = (5$0, IR[5:15] ! ^5$1, IR[5:15]) *
            (^IR[5], IR[5]);
     OBUS = ABUS; MD <= OBUS.
```

```
9.  ABUS = MD;
    BBUS = (AC ! IX) * (^IR[4] ! IR[4]);
    OBUS = ADD28(ABUS;BBUS;0);
    AC * ^IR[4] <= OBUS;
    IX *  IR[4] <= OBUS;
    ZFF <= ^(+/OBUS);
    CFF <= ADD0(ABUS;BBUS;0);
    NFF <= OBUS[0];
    VFF <= (ABUS[0] & BBUS[0] & ^(ADD1(ABUS;BBUS;0)) +
           (ABUS[0] & BBUS[0] & (ADD1(ABUS;BBUS;0));
    => (2).

10.  ((^IR[2] & ^IR[3]) ! (^IR[2] & IR[3]) !
      (IR[2] & IR[3]) ! (IR[2] & ^IR[3])  )
       / (14,11,12,13).

11. => ((IR[4] @ VFF) ! ^(IR[4] @ VFF)) / (14,2).

12. => ((IR[4] @ NFF) ! ^(IR[4] @ NFF)) / (14,2).

13. => (^(IR[4] @ ZFF)) / (2).

14. ABUS = (17$9, IR[5:15] ! ^17$1, IR[5:15]) *
           (^IR[5],IR[5]);
    BBUS = PC;
    OBUS = ADD28(ABUS;BBUS;0);
    PC <= OBUS;
    => (2).

15. OBUS = 28$0; CSR <= OBUS; TEMP <= OBUS; TIME <= OBUS; VI7 <= OBUS;
    VO7  <= OBUS; VI6  <= OBUS; VO6  <= OBUS; VI5 <= OBUS; VO5 <= OBUS;
    VI4  <= OBUS; VO4  <= OBUS; VI3  <= OBUS; VO3 <= OBUS; VI2 <= OBUS;
    VO2  <= OBUS; VOUT <= OBUS; SI8  <= OBUS; SO8 <= OBUS; SI7 <= OBUS;
    SO7  <= OBUS; SI6  <= OBUS; SO6  <= OBUS; SI5 <= OBUS; SO5 <= OBUS;
    SI4  <= OBUS; SI4  <= OBUS; SO4  <= OBUS; SI3 <= OBUS; SO3 <= OBUS;
    SI2  <= OBUS; SO2  <= OBUS; SOUT <= OBUS; FI4 <= OBUS; FO4 <= OBUS;
    FI3  <= OBUS; FO3  <= OBUS; FI2  <= OBUS; FO2 <= OBUS;
    FOUT <= OBUS; VSUM <= OBUS; VAP  <= OBUS; FSUM <= OBUS;
    FAP  <= OBUS; SSUM <= OBUS; SAP  <= OBUS.

16. BBUS = SMP; OBUS = BBUS; COUNT <= OBUS.

17. START = \1\; COUNT <= DEC(COUNT); ABUS=\01111100001000111101011100000\;
    BBUS = TIME; OBUS = FLADD(ABUS;BBUS); TIME <= OBUS;
    => (&/COUNT) / (2).

18. NO DELAY
    => (^CSR[13])/(62).

19. ABUS = VSF; BBUS = VIN; OBUS = FMUL(ABUS;BBUS); VIN <= OBUS;
    => (VFF) / (2).
```

```
20. ABUS = \01111110000010001111111011001\;
    BBUS = VIN; OBUS = FMUL(ABUS;BBUS); MD <= OBUS;
    => (VFF) / (2).

21. ABUS = \00000001001111111100010100000\;
    BBUS = VOUT; OBUS = FMUL(ABUS;BBUS); TEMP <= OBUS;
    => (VFF) / (2).

22. ABUS = MD; BBUS = TEMP; OBUS = FLADD(ABUS;BBUS); MD <= OBUS.

23. ABUS = \11111111001100101111111101101\;
    BBUS = VI2; OBUS = FMUL(ABUS;BBUS); TEMP <= OBUS;
    => (VFF) / (2).

24. ABUS = MD; BBUS = TEMP; OBUS = FLADD(ABUS;BBUS); MD <= OBUS.

25. ABUS = \10000001111101101101010011111\;
    BBUS = VO2; OBUS = FMUL(ABUS;BBUS); TEMP <= OBUS;
    => (VFF) / (2).

26. ABUS = MD; BBUS = TEMP; OBUS = FLADD(ABUS;BBUS); MD <= OBUS.

27. ABUS = \01111111110001000110010111110\;
    BBUS = VI3; OBUS = FMUL(ABUS;BBUS); TEMP <= OBUS;
    => (VFF) / (2).

28. ABUS = MD; BBUS = TEMP; OBUS = FLADD(ABUS;BBUS); MD <= OBUS.

29. ABUS = \00000010001100011000101101000\;
    BBUS = VO3; OBUS = FMUL(ABUS;BBUS); TEMP <= OBUS;
    => (VFF) / (2).

30. ABUS = MD; BBUS = TEMP; OBUS = FLADD(ABUS;BBUS); MD <= OBUS.

31. ABUS = \11111111111001111101100101010\;
    BBUS = VI4; OBUS = FMUL(ABUS;BBUS); TEMP <= OBUS;
    => (VFF) / (2).

32. ABUS = MD; BBUS = TEMP; OBUS = FLADD(ABUS;BBUS); MD <= OBUS.

33. ABUS = \10000010000110011111100011010\;
    BBUS = VO4; OBUS = FMUL(ABUS;BBUS); TEMP <= OBUS;
    => (VFF) / (2).

34. ABUS = MD; BBUS = TEMP; OBUS = FLADD(ABUS;BBUS); MD <= OBUS.

35. ABUS = \01111111100110110111110000000\;
    BBUS = VI5; OBUS = FMUL(ABUS;BBUS); TEMP <= OBUS;
    => (VFF) / (2).

36. ABUS = MD; BBUS = TEMP; OBUS = FLADD(ABUS;BBUS); MD <= OBUS.
```

```
37. ABUS = \00000001101000001111000010100\;
    BBUS = VO5; OBUS = FMUL(ABUS;BBUS); TEMP <= OBUS;
    => (VFF) / (2).

38. ABUS = MD; BBUS = TEMP; OBUS = FLADD(ABUS;BBUS); MD <= OBUS.


39. ABUS = \11111110111000001111100010010\;
    BBUS = VI6; OBUS = FMUL(ABUS;BBUS); TEMP <= OBUS;
    => (VFF) / (2).

40. ABUS = MD; BBUS = TEMP; OBUS = FLADD(ABUS;BBUS); MD <= OBUS.

41. ABUS = \10000000101110111010011111011\;
    BBUS = VO6; OBUS = FMUL(ABUS;BBUS); TEMP <= OBUS;
    => (VFF) / (2).

42. ABUS = MD; BBUS = TEMP; OBUS = FLADD(ABUS;BBUS); MD <= OBUS.

43. ABUS = \01111101100010010110101111011\;
    BBUS = VI7; OBUS = FMUL(ABUS;BBUS); TEMP <= OBUS;
    => (VFF) / (2).

44. ABUS = MD; BBUS = TEMP; OBUS = FLADD(ABUS;BBUS); MD <= OBUS.

45. ABUS = \01111111001111000111001001O1\;
    BBUS = VO7; OBUS = FMUL(ABUS;BBUS); TEMP <= OBUS;
    => (VFF) / (2).

46. ABUS = MD; BBUS = TEMP; OBUS = FLADD(ABUS;BBUS); MD <= OBUS;
    VOUT <= OBUS; TEMP <= OBUS.

47 ABUS = MD; BBUS = TEMP; OBUS = FMUL(ABUS;BBUS); MD <= OBUS;
    => (VFF) / (2).

48. ABUS = MD; BBUS = VSUM; OBUS = FLADD(ABUS;BBUS); VSUM <= OBUS;
    MD <= OBUS.

49. ABUS = MD; BBUS = TIME; OBUS = FLDIV(ABUS;BBUS); VAP <= OBUS.

50. BBUS = VI6; OBUS = BBUS; VI7 <= OBUS.

51. BBUS = VI5; OBUS = BBUS; VI6 <= OBUS.

52. BBUS = VI4; OBUS = BBUS; VI5 <= OBUS.

53. BBUS = VI3; OBUS = BBUS; VI4 <= OBUS.

54. BBUS = VI2; OBUS = BBUS; VI3 <= OBUS.

55. BBUS = VIN; OBUS = BBUS; VI2 <= OBUS.

56. BBUS = VO6; OBUS = BBUS; VO7 <= OBUS.
```

```
57. BBUS = VO5; OBUS = BBUS; VO6 <= OBUS.

58. BBUS = VO4; OBUS = BBUS; VO5 <= OBUS.

59. BBUS = VO3; OBUS = BBUS; VO4 <= OBUS.

60. BBUS = VO2; OBUS = BBUS; VO3 <= OBUS.

61. BBUS = VOUT; OBUS = BBUS; VO2 <= OBUS.

62. => (^CSR[14])/(88).

63. ABUS = FSF; BBUS = FIN; OBUS = FMUL(ABUS;BBUS); FIN <= OBUS;
    => (VFF) / (2).

64. ABUS = \01111011101001011100111001O1\;
    BBUS = FIN; OBUS = FMUL(ABUS;BBUS); MD <= OBUS;
    => (VFF) / (2).

65. ABUS = \00000000110111011011111110100\;
    BBUS = FOUT; OBUS = FMUL(ABUS;BBUS); TEMP <= OBUS;
    => (VFF) / (2).

66. ABUS = MD; BBUS = TEMP; OBUS = FLADD(ABUS;BBUS); MD <= OBUS.

67. ABUS = \11111011010010001000101001O0\;
    BBUS = FI2; OBUS = FMUL(ABUS;BBUS); TEMP <= OBUS;
    => (VFF) / (2).

68. ABUS = MD; BBUS = TEMP; OBUS = FLADD(ABUS;BBUS); MD <= OBUS.

69. ABUS = \10000001000011101100001111O0\;
    BBUS = FO2; OBUS = FMUL(ABUS;BBUS); TEMP <= OBUS;
    => (VFF) / (2).

70. ABUS = MD; BBUS = TEMP; OBUS = FLADD(ABUS;BBUS); MD <= OBUS.

71. ABUS = \11111011100111100010001101l0\;
    BBUS = FI3; OBUS = FMUL(ABUS;BBUS); TEMP <= OBUS;
    => (VFF) / (2).

72. ABUS = MD; BBUS = TEMP; OBUS = FLADD(ABUS;BBUS); MD <= OBUS.

73. ABUS = \00000000101000011001011111111\;
    BBUS = FO3; OBUS = FMUL(ABUS;BBUS); TEMP <= OBUS;
    => (VFF) / (2).

74. ABUS = MD; BBUS = TEMP; OBUS = FLADD(ABUS;BBUS); MD <= OBUS.

75. ABUS = \01111011001110011011101010l0\;
    BBUS = FI4; OBUS = FMUL(ABUS;BBUS); TEMP <= OBUS;
    => (VFF) / (2).
```

```
76. ABUS = MD; BBUS = TEMP; OBUS = FLADD(ABUS;BBUS); MD <= OBUS.

77. ABUS = \11111111100001110011111101010\;
    BBUS = FO4; OBUS = FMUL(ABUS;BBUS); TEMP <= OBUS;
    => (VFF) / (2).

78. ABUS = MD; BBUS = TEMP; OBUS = FLADD(ABUS;BBUS); MD <= OBUS;
    TEMP <= OBUS.

79. ABUS = MD; BBUS = TEMP; OBUS = FMUL(ABUS;BBUS); MD <= OBUS;
    => (VFF) / (2).

80. ABUS = MD; BBUS = VSUM; OBUS = FLADD(ABUS;BBUS); VSUM <= OBUS;
    MD <= OBUS.

81. ABUS = MD; BBUS = TIME; OBUS = FLDIV(ABUS;BBUS); VAP <= OBUS.

82. BBUS = FI3; OBUS = BBUS; FI4 <= OBUS.

83. BBUS = FI2; OBUS = BBUS; FI3 <= OBUS.

84. BBUS = FIN; OBUS = BBUS; FI2 <= OBUS.

85. BBUS = FO3; OBUS = BBUS; FO4 <= OBUS.

86. BBUS = FO2; OBUS = BBUS; FO3 <= OBUS.

87. BBUS = FOUT; OBUS = BBUS; FO2 <= OBUS.

88. => (^CSR[15])/(2).

89. ABUS = SSF; BBUS = SIN; OBUS = FMUL(ABUS;BBUS); SIN <= OBUS;
    => (VFF) / (2).

90. ABUS = \01111100000110110100000010011\;
    BBUS = SIN; OBUS = FMUL(ABUS;BBUS); MD <= OBUS;
    => (VFF) / (2).

91. ABUS = \00000001010111110101010011101\;
    BBUS = SOUT; OBUS = FMUL(ABUS;BBUS); TEMP <= OBUS;
    => (VFF) / (2).

92. ABUS = MD; BBUS = TEMP; OBUS = FLADD(ABUS;BBUS); MD <= OBUS.

93. ABUS = \11111101001101001100110000010\;
    BBUS = SI2; OBUS = FMUL(ABUS;BBUS); TEMP <= OBUS;
    => (VFF) / (2).

94. ABUS = MD; BBUS = TEMP; OBUS = FLADD(ABUS;BBUS); MD <= OBUS.

95. ABUS = \10000010001010001111111011111\;
    BBUS = SO2; OBUS = FMUL(ABUS;BBUS); TEMP <= OBUS;
    => (VFF) / (2).
```

```
96. ABUS = MD; BBUS = TEMP; OBUS = FLADD(ABUS;BBUS); MD <= OBUS.

97. ABUS = \01111101100110010011101100l1\;
    BBUS = SI3; OBUS = FMUL(ABUS;BBUS); TEMP <= OBUS;
    => (VFF) / (2).

98. ABUS = MD; BBUS = TEMP; OBUS = FLADD(ABUS;BBUS); MD <= OBUS.

99. ABUS = \00000010100100001001011010O0\;
    BBUS = SO3; OBUS = FMUL(ABUS;BBUS); TEMP <= OBUS;
    => (VFF) / (2).

100. ABUS = MD; BBUS = TEMP; OBUS = FLADD(ABUS;BBUS); MD <= OBUS.

101. ABUS = \11111101001110111001100011OO\;
     BBUS = SI4; OBUS = FMUL(ABUS;BBUS); TEMP <= OBUS;
    => (VFF) / (2).

102. ABUS = MD; BBUS = TEMP; OBUS = FLADD(ABUS;BBUS); MD <= OBUS.

103. ABUS = \10000010100110001000111101O1\;
     BBUS = SO4; OBUS = FMUL(ABUS;BBUS); TEMP <= OBUS;
    => (VFF) / (2).

104. ABUS = MD; BBUS = TEMP; OBUS = FLADD(ABUS;BBUS); MD <= OBUS.

105. ABUS = \11111100100010110100001110O1\;
     BBUS = SI5; OBUS = FMUL(ABUS;BBUS); TEMP <= OBUS;
    => (VFF) / (2).

106. ABUS = MD; BBUS = TEMP; OBUS = FLADD(ABUS;BBUS); MD <= OBUS.

107. ABUS = \00000010010010100111111011l1\;
     BBUS = SO5; OBUS = FMUL(ABUS;BBUS); TEMP <= OBUS;
    => (VFF) / (2).

108. ABUS = MD; BBUS = TEMP; OBUS = FLADD(ABUS;BBUS); MD <= OBUS.

109. ABUS = \01111101000111111011111001l1\;
     BBUS = SI6; OBUS = FMUL(ABUS;BBUS); TEMP <= OBUS;
    => (VFF) / (2).

110. ABUS = MD; BBUS = TEMP; OBUS = FLADD(ABUS;BBUS); MD <= OBUS.

111. ABUS = \10000001101001000010000011O0\;
     BBUS = SO6; OBUS = FMUL(ABUS;BBUS); TEMP <= OBUS;
    => (VFF) / (2).

112. ABUS = MD; BBUS = TEMP; OBUS = FLADD(ABUS;BBUS); MD <= OBUS.

113. ABUS = \11111100101000101011000101l0\;
     BBUS = SI7; OBUS = FMUL(ABUS;BBUS); TEMP <= OBUS;
    => (VFF) / (2).
```

```
114. ABUS = MD; BBUS = TEMP; OBUS = FLADD(ABUS;BBUS); MD <= OBUS.

115. ABUS = \00000000100100110010001011O1\;
     BBUS = SO7; OBUS = FMUL(ABUS;BBUS); TEMP <= OBUS;
    => (VFF) / (2).

117. ABUS = MD; BBUS = TEMP; OBUS = FLADD(ABUS;BBUS); MD <= OBUS.

118. ABUS = \01111011011000011101011001O0\;
     BBUS = SI8; OBUS = FMUL(ABUS;BBUS); TEMP <= OBUS;
    => (VFF) / (2).

119. ABUS = MD; BBUS = TEMP; OBUS = FLADD(ABUS;BBUS); MD <= OBUS.

120. ABUS = \11111110110110111011010110O1\;
     BBUS = SO8; OBUS = FMUL(ABUS;BBUS); TEMP <= OBUS;
    => (VFF) / (2).

121. ABUS = MD; BBUS = TEMP; OBUS = FLADD(ABUS;BBUS); MD <= OBUS;
    SOUT <= OBUS; TEMP <= OBUS.

122. ABUS = MD; BBUS = TEMP; OBUS = FMUL(ABUS;BBUS); MD <= OBUS;
    => (VFF) / (2).

123. ABUS = MD; BBUS = VSUM; OBUS = FLADD(ABUS;BBUS); SSUM <= OBUS;
    MD <= OBUS.

124. ABUS = MD; BBUS = TIME; OBUS = FLDIV(ABUS;BBUS); SAP <= OBUS.

125. BBUS = SI7; OBUS = BBUS; SI8 <= OBUS.

126. BBUS = SI6; OBUS = BBUS; SI7 <= OBUS.

127. BBUS = SI5; OBUS = BBUS; SI6 <= OBUS.

128. BBUS = SI4; OBUS = BBUS; SI5 <= OBUS.

129. BBUS = SI3; OBUS = BBUS; SI4 <= OBUS.

130. BBUS = SI2; OBUS = BBUS; SI3 <= OBUS.

131. BBUS = SIN; OBUS = BBUS; SI2 <= OBUS.

132. BBUS = SO7; OBUS = BBUS; SO8 <= OBUS.

133. BBUS = SO6; OBUS = BBUS; SO7 <= OBUS.

134. BBUS = SO5; OBUS = BBUS; SO6 <= OBUS.

135. BBUS = SO4; OBUS = BBUS; SO5 <= OBUS.

136. BBUS = SO3; OBUS = BBUS; SO4 <= OBUS.
```

```
137. BBUS = SO2; OBUS = BBUS; SO3 <= OBUS.

138. BBUS = SOUT; OBUS = BBUS; SO2 <= OBUS;
     => (17).

139. => (IR[1] & IR[2]) / (142).

140. MD <= BUSFN(M;DCD(IX)).

141. ABUS = MD;
     OBUS = ABUS;
     OUTPUT <= OBUS;
     => (2).

142. => (IR[13]) / (144).

143. MD <= BUSFN(M;DCD(IX));
     PC <= INC(PC);
     => (146).

144. => (^IR[15]) / (146).

145. BBUS = (AC ! IX) * (^IR[14], IR[14]);
     OBUS = BBUS;
     MD <= OBUS.

146. => (IR[4]) / (149).

147. NO DELAY
     => (^IR[6]) / (2).

148. ABUS = MD;
     OBUS = ABUS;
     AC * ^IR[5] <= OBUS;
     IX *  IR[5] <= OBUS;
     => (2).

149. ABUS = (22$0, IR[7:12]);
     BBUS = 28$32;
     OBUS = ADD(ABUS;BBUS;0);
     MA <= OBUS.

150. M*DCD(MA) <= MD;
     => (2).

151. NO DELAY
     => ((IR[4] & IR[5]) ! (IR[4] & ^IR[5]) / (153,154).

152. ABUS = (6$0, IR[6:15] ! ^(6$1), IR[6:15]) *
            (^IR[6],IR[6]);
     BBUS = AC;
     => (155).
```

```
153. ABUS = MD;
     BBUS = AC;
     => (155).

154. BBUS = IX;
     OBUS = BBUS;
     MD <= OBUS;
     => (153).

155. OBUS = ADD28(ABUS;BBUS;0);
     ZFF <= ^(+/OBUS);
     CFF <= ADD0(ABUS;BBUS;0);
     NFF <= OBUS[0];
     VFF <= (ABUS[0] & BBUS[0] & ^(ADD1(ABUS;BBUS;0)) +
            (ABUS[0] & BBUS[0] & ADD1(ABUS;BBUS:0);
     => (2).



END OF SEQUENCE

  DATAOUT = ( SOUT ! VOUT ! FOUT ) *
            ( ^SELECT[0]& ^SELECT[1], ^SELECT[0] & SELECT[1], SELECT[0] ).

END.



CLUNIT: FLADD(X,Y)
  INPUTS: X[28];Y[28].
  OUTPUTS: FLADD[28].
  BODY
    FLADD = X[28].
  END.


CLUNIT: FLMULT(X,Y)
  INPUTS: X[28];Y[28].
  OUTPUTS: FLMULT[28].
  BODY
    FLMULT = X[28].
  END.
```

```
CLUNIT: FLDIV(X,Y)
  INPUTS: X[28];Y[28].
  OUTPUTS: FLDIV[28].
  BODY
    FLDIV = X[28].
  END.


CLUNIT: FULLADD(X;Y;CIN)
  INPUTS: X;Y;CIN.
  OUTPUTS: FULLADD[2].
  CTERMS: A;B;C;SUM;COUT.
  BODY
    A = X @ Y;
    B = X & Y;
    SUM = A @ CIN;
    C = A & C;
    FULLADD[0] = COUT;
    FULLADD[1] = SUM.
  END.



CLUNIT:  ADDER(XIN;YIN;CIN) {I}
  INPUTS: XIN[I];YIN[I];CIN.
  OUTPUTS: SUMOUT[I+1].
  CLUNITS: FA[2] <: FULLADD.
  CTERMS: CARRY[I];SUM[I].
  BODY
    CARRY[I-1],SUM[I-1] = FA(XIN[I-1];YIN[I-1];CIN);
    FOR J = I-2 TO 0 CONSTRUCT
      CARRY[J],SUM[J] = FA(SIN[J];YIN[J];CARRY[J+1])
    ROF;
    SUMOUT = CARRY[0],SUM.
  END.


CLUNIT: INC(X)
  INPUTS: X[28].
  OUTPUTS: TERMOUT[28].
  CTERMS: TA[28].
  BODY
    FOR J = 27 TO 0 CONSTRUCT
      IF J = 27 THEN
        TA[J] = 1
      ELSE
        TA[J] = X[J+1] & TA[J+1]
      FI
    ROF;
    FOR J = 27 TO 0 CONSTRUCT
      TERMOUT[J] = X[J] @ TA[J]
    ROF.
  END.
```

```
CLUNIT: DECODER(A)
  INPUTS: A[16].
  OUTPUTS: DCDOUT[65536].
  BODY
    FOR J = 0 TO 65535 CONSTRUCT
      DCDOUT[J] = TERM(J;A)
    ROF.
  END.


CLUNIT: BUSFN(MEM;F)
  INPUTS: MEM<65536>[28];F[28].
  OUTPUTS: WORDOUT[L].
  CTERMS: N<65536>[28].
  BODY
    FOR I = 0 TO 65535 CONSTRUCT
      N<I> = MEM<I> & F<I>
    ROF;
    WORDOUT = +//N.
  END.



CLUNIT: DEC(X)
  INPUTS: X[28].
  OUTPUTS: TERMOUT[28].
  CTERMS: TA[I]; CARRY[I]; TEMP[I].
  BODY
    FOR J=0 TO 27 CONSTRUCT
      TA[I] = ^X[I]
    ROF;
    CARRY[27] = 1;
    FOR J = 26 TO 0 CONSTRUCT
      CARRY[J] = CARRY[J-1] & TA[J-1]
    ROF;
    FOR J = 0 TO 27 CONSTRUCT
      TEMP[J] = TA[J] @ CARRY[J]
    ROF;
    FOR J = 0 TO 27 CONSTRUCT
      TERMOUT[J] = ^TEMP[J]
    ROF.
  END.
```

DISTRIBUTION LIST

	Copies
Commander Defense Technical Information Center Bldg. 5, Cameron Station ATTN: DDAC Alexandria, VA 22304-9990	12
Commander U.S. Army Test-Evaluation Command ATTN: AMSTE-TA-R Aberdeen Proving Ground, MY 21005-5055	2
Manager Defense Logistics Studies Information Exchange ATTN: AMXMC-D Fort Lee, VA 23801-6044	2
Commander U.S. Army Tank-Automotive Command ATTN: ASQNC-TAC-DIT (Technical Library) AMSTA-CF (Dr. Oscar) AMSTA-RYA (Mr. Janosi) AMSTA-RY (Dr. Beck) AMSTA-RYA (Mr. Reid) AMSTA-RYA (Mr. Hudas) Warren, MI 48397-5000	2 1 1 1 15 15
Director U.S. Material Systems Analysis Activity ATTN: AMXSY-MP (Mr. Cohen) Aberdeen Proving Ground, MD 21005-5071	1